



Xamarin




e-Portfolio

Tobias Bühler
TINF18B3

Software-Engineering I
02.11.2019

INHALT DES VORTRAGS

- ▶ Was ist Xamarin?
 - ▶ Was sind Cross-Plattform-Apps?
 - ▶ Vorteile und Nachteile?
 - ▶ Xamarin-Forms
 - ▶ App-Aufbau
 - ▶ Wichtige Elemente
 - ▶ Page und Page-Class
 - ▶ Aufgabe
- 
- Several white lines of varying lengths and orientations are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

iOS



Platform-Specific C#

Native iOS App

Native Android App

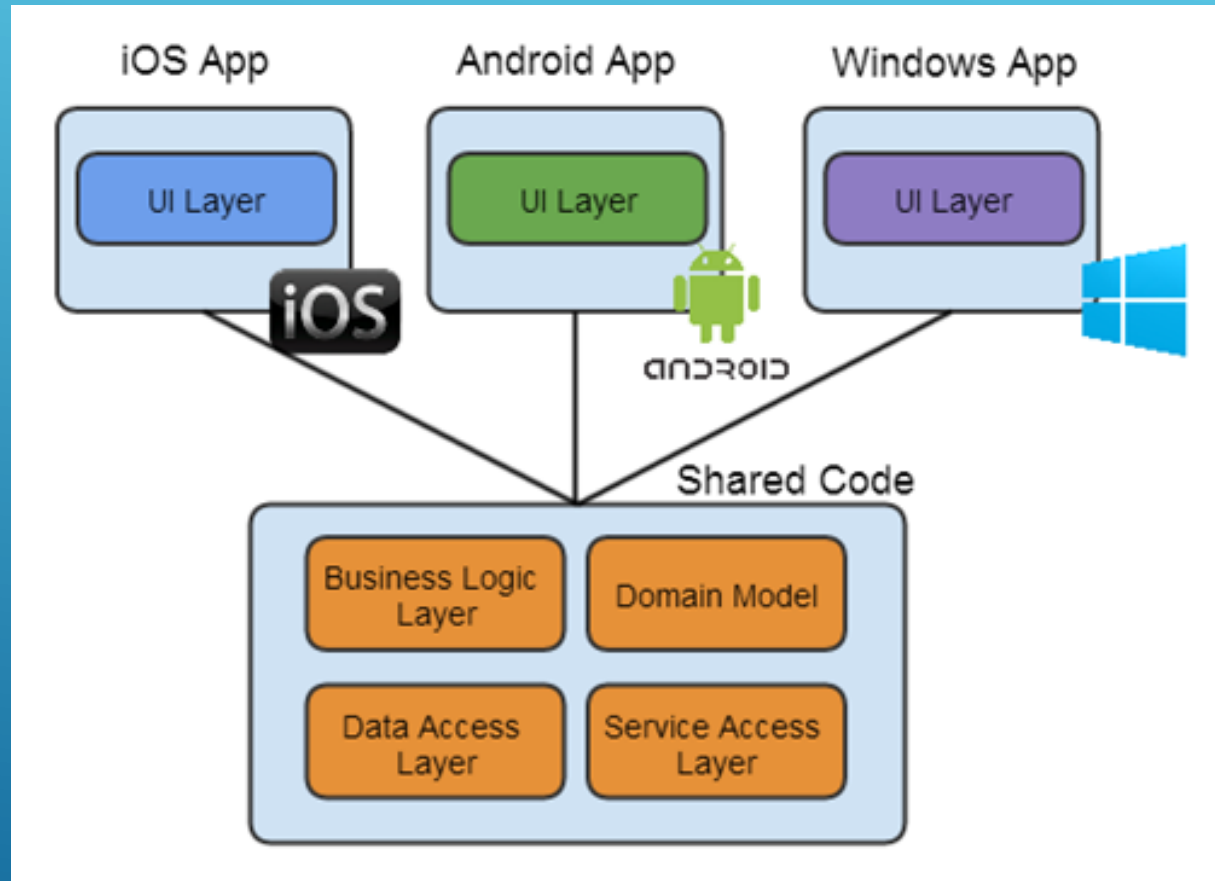
Native Windows Phone App

Shared C# User Interface Code

Shared C# App Logic

WAS IST XAMARIN?

Mit den Komponenten *Xamarin.iOS* und *Xamarin.Android* ist es möglich, native Apps für iOS, Android und Windows in C# entweder mit Xamarin Studio oder Visual Studio zu entwickeln.



WAS SIND CROSS-PLATFORM-APPS?

Dies sind Apps welche nicht ausschließlich für eine Plattform festgelegt sind.

Die Basis ist ein gemeinsames Backend, welches von allen übergeordneten UI Layer der unterschiedlichen Systeme genutzt wird.

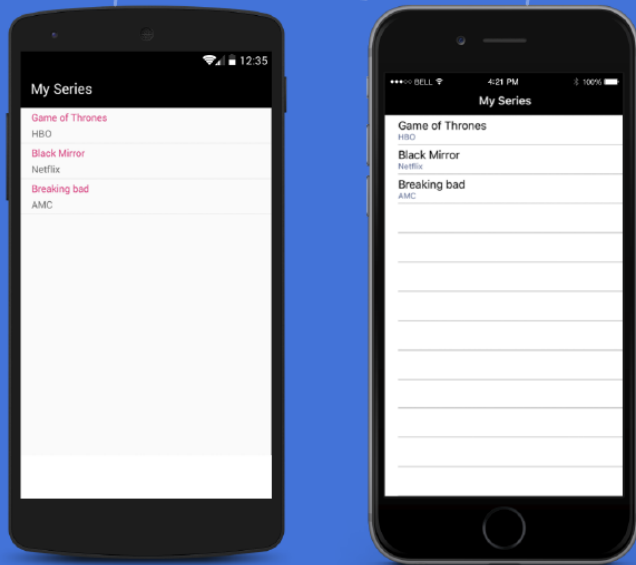
Vorteile

- ▶ Ein Projekt pro App und damit nur noch ein Entwicklerteam
- ▶ Schnellere plattformübergreifende Änderungsmöglichkeiten
- ▶ Wird stetig weiterentwickelt
- ▶ Visual Studio für die Entwicklung aller Plattformen nutzbar

Nachteile

- ▶ Bietet nicht alle nativen Funktionen der jeweiligen Plattformen direkt an
 - ▶ Komplex
- 
- Several white lines of varying lengths and orientations are positioned in the bottom right corner of the slide, creating a modern, abstract design element.

Xamarin forms



Don't
repeat
yourself

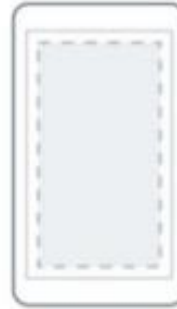
XAMARIN-FORMS

Applikation Framework um User Interfaces zu erstellen.

Alle Xamarin Formelemente werden später in native Formelemente der jeweiligen Plattform übersetzt.

App - Aufbau

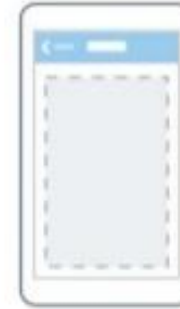
Xamarin.Forms Pages



ContentPage



MasterDetailPage



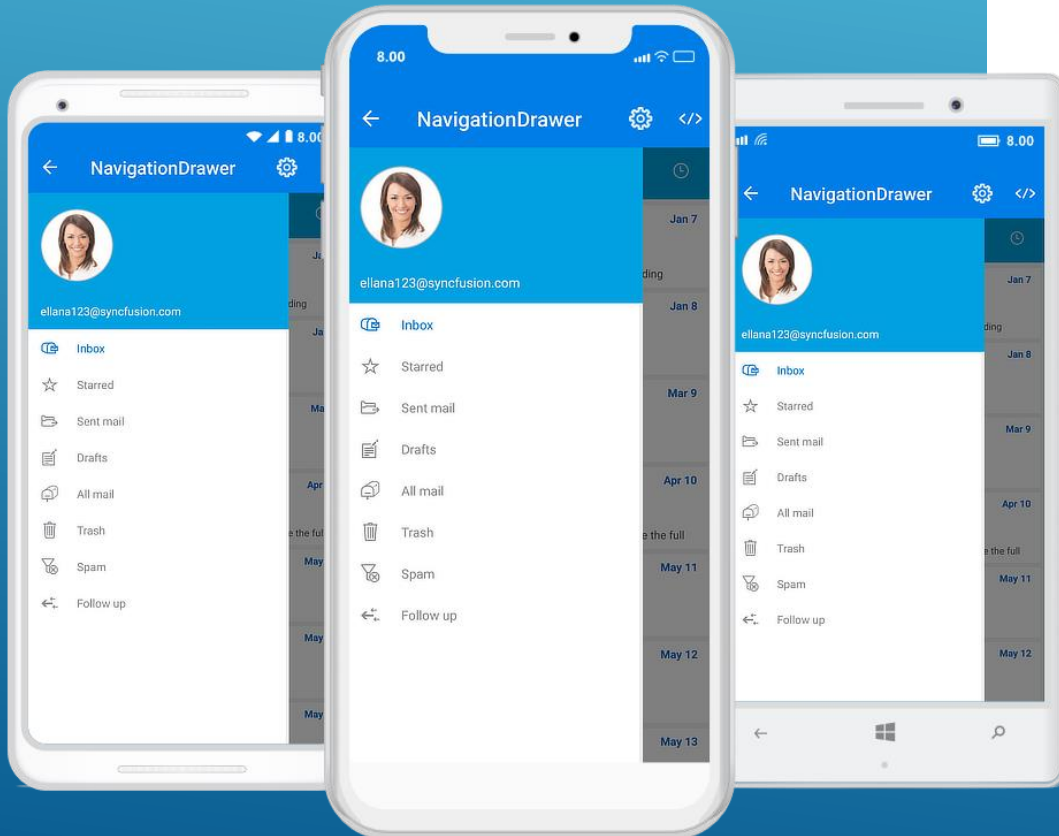
NavigationPage



TabbedPage



CarouselPage



WICHTIGE FORM ELEMENTE

- ▶ `<Button Name="btn" OnClick="Button_Clicked" Text="Button"/>`
- ▶ `<Entry Name="entry"/>`
- ▶ `<ListView />`
- ▶ `<Label Text="Ein Label"/>`
- ▶ `<Switch />`
- ▶ `<StackLayout />`
- ▶ `<ContentPage />`

Aufbau der Page

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="TodoChecker.TODOItemPage">
  <ContentPage.Content>
    <StackLayout Margin="20" VerticalOptions="StartAndExpand">
      <Label Text="Name" />
      <Entry Text="{Binding Name}" />
      <Label Text="Notes" />
      <Entry Text="{Binding Notes}" />
      <Label Text="Done" />
      <Switch IsToggled="{Binding Done}" />
      <Button Text="Save" Clicked="OnSaveClicked" />
      <Button Text="Delete" Clicked="OnDeleteClicked" />
      <Button Text="Cancel" Clicked="OnCancelClicked" />
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

Aufbau der zugehörigen Klasse

```
MainPage.xaml.cs
DefuseBomb

4   using System.Text;
5   using System.Threading.Tasks;
6   using Xamarin.Forms;
7
8   namespace DefuseBomb
9   {
10      public partial class MainPage : ContentPage
11      {
12
13
14      public MainPage()
15      {
16          InitializeComponent();
17      }
18
19  }
```

AUFGABE BOMB-DEFUSE

- ▶ Erstellt ein neues Cross-Platform-App Projekt
- ▶ Füge auf der MainPage (MainPage.xaml) Elemente hinzu (2 Label / 2 Buttons)
- ▶ Fügt in der MainPage Klasse (MainPage.xaml.cs) eine OnClick Methode für die Buttons hinzu.
- ▶ Durch Drücken der Buttons soll versucht werden die richtige Kombination zu erraten.
- ▶ Der nächste benötigte Wert des Codes wird zufällig beim Starten der App sowie nach jedem gedrückten Button erzeugt.
- ▶ Der Code (und damit ein Spieldurchgang) hat 3 Stellen.
- ▶ Bei einer explodierten oder entschärften Bombe soll dementsprechend eine Nachricht angezeigt werden mit der Möglichkeit ein neues Spiel zu starten.